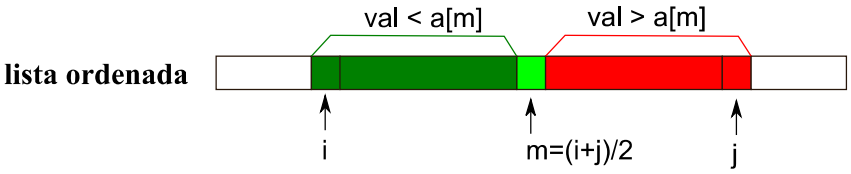
Busca Binária

Ordenação é útil para humanos

- Organizamos dados
- Facilitamos a busca
- Permitem lidar com volumen grande de dados -- imagine um dicionário português sem ordenação

Ordenação é também útil para projetar algoritmos eficientes. Um exemplo é a busca binária onde com apenas uma comparação podemos diminuir o tamanho do espaço de busca pela metade.



Se temos uma lista `a` ordenada e estamos buscando por um valor `val` numa porção da lista delimitada pelos índicies `i` e `j`, podemos comparar `val` com o elemento no meio da porção `a[m]` e então

- se `val < a[m]`, a busca poderá ser restrita à porção entre os índices `i` e `m-1`, pois certamente `val` não estará à direita de `a[m]`,
- similarmente, se `val > a[m]`, a busca poderá ser restrita à porção entre os índices `m+1` e `j`

invariante: se `val in a` então `val in a[i:j+1]`

Ou, em outras palavras

invariante: `not val in a[:i]` e `not val in a[j+1:]`

In [8]:

```
def buscaBinaria(val, a):
    return buscaBinariaEm(0, len(a)-1, val, a)

def buscaBinariaEm(i, j, val, a):
    m = (i + j) // 2
    while i <= j and val != a[m]:
        if val < a[m]:
            j = m-1
        else:
            i = m+1
        m = (i + j) // 2
    return i<=j

xs = list(range(1,1000,3))
buscaBinaria(106, xs)
```

Out[8]:

```
True
```